Static Program Analysis

Lecture 3: Dataflow Analysis II (Live Variables Analysis)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/spa11/

Summer Semester 2011

Outline

Labeled Programs

- Goal: localization of analysis information
- Dataflow information will be associated with
  - `skip` statements
  - assignments
  - tests in conditionals (`if`) and loops (`while`)
- Assume set of labels L with meta variable $l \in L$ (usually $L = \mathbb{N}$)

Definition (Labeled WHILE programs)

The syntax of labeled WHILE programs is defined by the following context-free grammar:

$$\begin{aligned} a &::= z \mid x \mid a_1 \texttt{+} a_2 \mid a_1 \texttt{-} a_2 \mid a_1 \texttt{*} a_2 \in AExp \\ b &::= t \mid a_1 \texttt{=} a_2 \mid a_1 \texttt{>} a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp \\ c &::= [\texttt{skip}]^l \mid [x \texttt{ := } a]^l \mid c_1 \texttt{;} c_2 \mid \\ &\qquad \texttt{if } [b]^l \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } [b]^l \texttt{ do } c \in Cmd \end{aligned}$$

- All labels in $c \in Cmd$ assumed distinct, denoted by L_c
- Labeled fragments of c called blocks, denoted by Blk_c

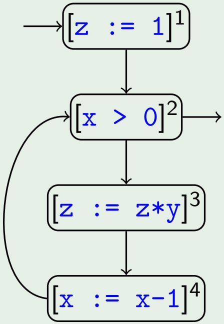
Representing Control Flow

Example

$c = $ `[z := 1]`^{1}`;`
```
    while [x > 0]^2 do
      [z := z*y]^3;
      [x := x-1]^4
```

$$\begin{aligned} \mathsf{init}(c) &= 1 \\ \mathsf{final}(c) &= \{2\} \\ \mathsf{flow}(c) &= \{(1,2),(2,3),(3,4),(4,2)\} \end{aligned}$$

Visualization by (control) flow graph:


$$\begin{tikzcd} \arrow[r] & {[\texttt{z := 1}]^1} \arrow[d] & \\ & {[\texttt{x > 0}]^2} \arrow[d] \arrow[r] & {} \\ & {[\texttt{z := z*y}]^3} \arrow[d] & \\ & {[\texttt{x := x-1}]^4} \arrow[uu, bend left=60] & \end{tikzcd}$$

Goal of Available Expressions Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- can be used to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example (Available Expressions Analysis)

```
[x := a+b]^1;
[y := a*b]^2;
while [y > a+b]^3 do
  [a := a+1]^4;
  [x := a+b]^5
```

- a+b available at label 3
- a+b not available at label 5
- possible optimization:
  `while [y > x]^3 do`

The Equation System

Reminder:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

Example (AE equation system)

$c = [\texttt{x := a+b}]^1;$
$\quad [\texttt{y := a*b}]^2;$
$\quad \texttt{while } [\texttt{y > a+b}]^3 \texttt{ do}$
$\qquad [\texttt{a := a+1}]^4;$
$\qquad [\texttt{x := a+b}]^5$

$l \in L_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\texttt{a+b}\}$
2	$\emptyset$	$\{\texttt{a*b}\}$
3	$\emptyset$	$\{\texttt{a+b}\}$
4	$\{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\texttt{a+b}\}$

Equations:

$$\begin{aligned} \mathsf{AE}_1 &= \emptyset \\ \mathsf{AE}_2 &= \varphi_1(\mathsf{AE}_1) = \mathsf{AE}_1 \cup \{\texttt{a+b}\} \\ \mathsf{AE}_3 &= \varphi_2(\mathsf{AE}_2) \cap \varphi_5(\mathsf{AE}_5) \\ &= (\mathsf{AE}_2 \cup \{\texttt{a*b}\}) \cap (\mathsf{AE}_5 \cup \{\texttt{a+b}\}) \\ \mathsf{AE}_4 &= \varphi_3(\mathsf{AE}_3) = \mathsf{AE}_3 \cup \{\texttt{a+b}\} \\ \mathsf{AE}_5 &= \varphi_4(\mathsf{AE}_4) = \mathsf{AE}_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\} \end{aligned}$$

Solution:

$$\begin{aligned} \mathsf{AE}_1 &= \emptyset \\ \mathsf{AE}_2 &= \{\texttt{a+b}\} \\ \mathsf{AE}_3 &= \{\texttt{a+b}\} \\ \mathsf{AE}_4 &= \{\texttt{a+b}\} \\ \mathsf{AE}_5 &= \emptyset \end{aligned}$$

Outline

Goal of Live Variables Analysis

Live Variables Analysis

The goal of Live Variables Analysis is to determine, for each program point, which variables *may* be live at the exit from the point.

- A variable is called live at the exit from a block if there exists a path from the block to a use of the variable that does not re-define the variable
- All variables considered to be live at the end of the program (alternative: restriction to output variables)
- Can be used for Dead Code Elimination: remove assignments to non-live variables

An Example

Example 3.1 (Live Variables Analysis)

```
[x := 2]^1;
[y := 4]^2;
[x := 1]^3;
if [y > 0]^4 then
  [z := x]^5
else
  [z := y*y]^6;
[x := z]^7
```

- x not live at exit from label 1
- y live at exit from 2
- x live at exit from 3
- z live at exits from 5 and 6
- possible optimization: remove $[\texttt{x := 2}]^1$

Formalizing Live Variables Analysis I

- A variable on the left-hand side of an assignment is killed by the assignment; tests and `skip` do not kill
- Formally: $\mathrm{kill}_{\mathrm{LV}} : \mathit{Blk}_c \to 2^{\mathit{Var}_c}$ is defined by

$$\begin{aligned}\mathrm{kill}_{\mathrm{LV}}([\texttt{skip}]^l) &:= \emptyset\\ \mathrm{kill}_{\mathrm{LV}}([x\ \texttt{:=}\ a]^l) &:= \{x\}\\ \mathrm{kill}_{\mathrm{LV}}([b]^l) &:= \emptyset\end{aligned}$$

- Every reading access generates a live variable
- Formally: $\mathrm{gen}_{\mathrm{LV}} : \mathit{Blk}_c \to 2^{\mathit{Var}_c}$ is defined by

$$\begin{aligned}\mathrm{gen}_{\mathrm{LV}}([\texttt{skip}]^l) &:= \emptyset\\ \mathrm{gen}_{\mathrm{LV}}([x\ \texttt{:=}\ a]^l) &:= \mathit{Var}_a\\ \mathrm{gen}_{\mathrm{LV}}([b]^l) &:= \mathit{Var}_b\end{aligned}$$

Formalizing Live Variables Analysis II

Example 3.2 ($\text{kill}_{\text{LV}}/\text{gen}_{\text{LV}}$ functions)

```
c = [x := 2]^1;
    [y := 4]^2;
    [x := 1]^3;
    if [y > 0]^4 then
      [z := x]^5
    else
      [z := y*y]^6;
    [x := z]^7
```

- $Var_c = \{x, y, z\}$
-

$l \in L_c$	$\text{kill}_{\text{LV}}(B^l)$	$\text{gen}_{\text{LV}}(B^l)$
1	$\{x\}$	$\emptyset$
2	$\{y\}$	$\emptyset$
3	$\{x\}$	$\emptyset$
4	$\emptyset$	$\{y\}$
5	$\{z\}$	$\{x\}$
6	$\{z\}$	$\{y\}$
7	$\{x\}$	$\{z\}$

The Equation System I

- For each $l \in L_c$, $\mathrm{LV}_l \subseteq \mathit{Var}_c$ represents the set of live variables at the exit of block B^l
- Formally, for a program $c \in \mathit{Cmd}$ with isolated exits:
$$\mathrm{LV}_l = \begin{cases} \mathit{Var}_c & \text{if } l \in \mathrm{final}(c) \\ \bigcup\{\varphi_{l'}(\mathrm{LV}_{l'}) \mid (l,l') \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$
where $\varphi_{l'} : 2^{\mathit{Var}_c} \to 2^{\mathit{Var}_c}$ denotes the transfer function of block $B^{l'}$, given by
$$\varphi_{l'}(V) := (V \setminus \mathrm{kill}_{\mathrm{LV}}(B^{l'})) \cup \mathrm{gen}_{\mathrm{LV}}(B^{l'})$$
- Characterization of analysis:

backward:	starts in $\mathrm{final}(c)$ and proceeds upwards
may:	$\bigcup$ in equation for LV_l
flow-sensitive:	results depending on order of assignments

- Later: solution not necessarily unique

  $\Longrightarrow$ choose least one

The Equation System II

Reminder:

$$\mathsf{LV}_l = \begin{cases} \mathit{Var}_c & \text{if } l \in \mathsf{final}(c) \\ \bigcup\{\varphi_{l'}(\mathsf{LV}_{l'}) \mid (l,l') \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(V) = (V \setminus \mathsf{kill}_{\mathsf{LV}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{LV}}(B^{l'})$$

Example 3.3 (LV equation system)

$c =$
```
[x := 2]^1;[y := 4]^2;
[x := 1]^3;
if [y > 0]^4 then
  [z := x]^5
else
  [z := y*y]^6;
[x := z]^7
```

$l \in L_c$	$\mathsf{kill}_{\mathsf{LV}}(B^l)$	$\mathsf{gen}_{\mathsf{LV}}(B^l)$
1	$\{x\}$	$\emptyset$
2	$\{y\}$	$\emptyset$
3	$\{x\}$	$\emptyset$
4	$\emptyset$	$\{y\}$
5	$\{z\}$	$\{x\}$
6	$\{z\}$	$\{y\}$
7	$\{x\}$	$\{z\}$

$$\begin{aligned}
\mathsf{LV}_1 &= \varphi_2(\mathsf{LV}_2) = \mathsf{LV}_2 \setminus \{y\} \\
\mathsf{LV}_2 &= \varphi_3(\mathsf{LV}_3) = \mathsf{LV}_3 \setminus \{x\} \\
\mathsf{LV}_3 &= \varphi_4(\mathsf{LV}_4) = \mathsf{LV}_4 \cup \{y\} \\
\mathsf{LV}_4 &= \varphi_5(\mathsf{LV}_5) \cup \varphi_6(\mathsf{LV}_6) \\
&= ((\mathsf{LV}_5 \setminus \{z\}) \cup \{x\}) \cup ((\mathsf{LV}_6 \setminus \{z\}) \cup \{y\}) \\
\mathsf{LV}_5 &= \varphi_7(\mathsf{LV}_7) = (\mathsf{LV}_7 \setminus \{x\}) \cup \{z\} \\
\mathsf{LV}_6 &= \varphi_7(\mathsf{LV}_7) = (\mathsf{LV}_7 \setminus \{x\}) \cup \{z\} \\
\mathsf{LV}_7 &= \{x, y, z\}
\end{aligned}$$

Solution:

$$\begin{aligned}
\mathsf{LV}_1 &= \emptyset \\
\mathsf{LV}_2 &= \{y\} \\
\mathsf{LV}_3 &= \{x, y\} \\
\mathsf{LV}_4 &= \{x, y\} \\
\mathsf{LV}_5 &= \{y, z\} \\
\mathsf{LV}_6 &= \{y, z\} \\
\mathsf{LV}_7 &= \{x, y, z\}
\end{aligned}$$

Outline

Similarities Between Analysis Problems

- **Observation:** the analyses presented so far have some similarities

$\Longrightarrow$ Look for underlying framework

- **Advantage:** possibility for designing (efficient) generic algorithms for solving dataflow equations
- **Overall pattern:** for $c \in Cmd$ and $l \in L_c$, the analysis information (AI) is described by equations of the form

$$\mathrm{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{\varphi_{l'}(\mathrm{AI}_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

where

- the set of extremal labels, E, is $\{\mathrm{init}(c)\}$ or $\mathrm{final}(c)$
- ι specifies the extremal analysis information
- the combination operator, $\bigsqcup$, is $\bigcap$ or $\bigcup$
- $\varphi_{l'}$ denotes the transfer function of block $B^{l'}$
- the flow relation F is $\mathrm{flow}(c)$ or $\mathrm{flow}^R(c)$ $(:= \{(l', l) \mid (l, l') \in \mathrm{flow}(c)\})$

Characterization of Analyses

- **Direction of information flow:**
  - forward:
    - $F = \text{flow}(c)$
    - AI_l concerns entry of B^l
    - c has isolated entry
  - backward:
    - $F = \text{flow}^R(c)$
    - AI_l concerns exit of B^l
    - c has isolated exits
- **Quantification over paths:**
  - may:
    - $\bigsqcup = \bigcup$
    - property satisfied by some path
    - interested in least solution (later)
  - must:
    - $\bigsqcup = \bigcap$
    - property satisfied by all paths
    - interested in greatest solution (later)

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterize solution of equation system as fixpoint of a transformation
2. Introduce partial order for comparing analysis results
3. Establish least upper bound as combination operator
4. Ensure monotonicity of transfer functions
5. Guarantee termination of fixpoint iteration by ascending chain condition
6. Optimize fixpoint iteration by worklist algorithm

Outline

Motivation

- **Wanted:** solution of (dataflow) equation system
- **Problem:** recursive dependencies between dataflow variables
- **Idea:** characterize solution as fixpoint of transformation:

$$(\mathsf{AI}_l = \tau_l)_{l \in L_c} \iff \Phi((\mathsf{AI}_l)_{l \in L_c}) = (\mathsf{AI}_l)_{l \in L_c}$$

  where $\Phi\left((\mathsf{AI}_l)_{l \in L_c}\right) := (\tau_l)_{l \in L_c}$
- **Approach:** approximate fixpoint by iteration

Partial Orders

The domain of analysis information usually forms a partial order where the ordering relation compares the “precision” of information.

Definition 3.4 (Partial order)

A partial order (PO) $(D, \sqsubseteq)$ consists of a set D, called domain, and of a relation $\sqsubseteq \subseteq D \times D$ such that, for every $d_1, d_2, d_3 \in D$,

reflexivity: $d_1 \sqsubseteq d_1$
transitivity: $d_1 \sqsubseteq d_2$ and $d_2 \sqsubseteq d_3 \implies d_1 \sqsubseteq d_3$
antisymmetry: $d_1 \sqsubseteq d_2$ and $d_2 \sqsubseteq d_1 \implies d_1 = d_2$

It is called total if, in addition, always $d_1 \sqsubseteq d_2$ or $d_2 \sqsubseteq d_1$.

Example 3.5

1. $(\mathbb{N}, \leq)$ is a total partial order
2. $(\mathbb{N}, <)$ is not a partial order (since not reflexive)
3. (Live Variables) $(2^{Var_c}, \subseteq)$ is a (non-total) partial order
4. (Available Expressions) $(2^{AExp_c}, \supseteq)$ is a (non-total) partial order

Upper Bounds

In the dataflow equation system, analysis information from several predecessors is combined by taking the least upper bound.

Definition 3.6 ((Least) upper bound)

Let $(D, \sqsubseteq)$ be a partial order and $S \subseteq D$.

1. An element $d \in D$ is called an upper bound of S if $s \sqsubseteq d$ for every $s \in S$ (notation: $S \sqsubseteq d$).
2. An upper bound d of S is called least upper bound (LUB) or supremum of S if $d \sqsubseteq d'$ for every upper bound d' of S (notation: $d = \bigsqcup S$).

Example 3.7

1. $S \subseteq \mathbb{N}$ has a LUB in $(\mathbb{N}, \leq)$ iff it is finite
2. (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$. Given $V_1, \ldots, V_n \subseteq Var_c$,
$$\bigsqcup\{V_1, \ldots, V_n\} = \bigcup\{V_1, \ldots, V_n\}$$
3. (Avail. Expr.) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$. Given $A_1, \ldots, A_n \subseteq AExp_c$,
$$\bigsqcup\{A_1, \ldots, A_n\} = \bigcap\{A_1, \ldots, A_n\}$$

Complete Lattices

Since $\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F\}$ can contain arbitrary elements, the existence of least upper bounds must be ensured for arbitrary subsets.

Definition 3.8 (Complete lattice)

A complete lattice is a partial order $(D, \sqsubseteq)$ such that all subsets of D have least upper bounds. In this case,

$$\bot := \bigsqcup \emptyset$$

denotes the least element of D.

Example 3.9

1. $(\mathbb{N}, \leq)$ is not a complete lattice as, e.g., $\mathbb{N}$ does not have a LUB
2. (Live Variables)
   $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$ is a complete lattice with $\bot = \emptyset$
3. (Available Expressions)
   $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$ is a complete lattice with $\bot = AExp_c$

Duality in Complete Lattices

- Dual concept of least upper bound: greatest lower bound
- **Definitions:**
  - An element $d \in D$ is called a lower bound of $S \subseteq D$ if $d \sqsubseteq s$ for every $s \in S$ (notation: $d \sqsubseteq S$).
  - A lower bound d is called greatest lower bound (GLB) or infimum of S if $d' \sqsubseteq d$ for every lower bound d' of S (notation: $d = \bigsqcap S$).
- **Examples:**
  - (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$, $\bigsqcap\{V_1, \ldots, V_n\} = \bigcap\{V_1, \ldots, V_n\}$
  - (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$, $\bigsqcap\{A_1, \ldots, A_n\} = \bigcup\{A_1, \ldots, A_n\}$
- **Lemma:** the following are equivalent:
  - $(D, \sqsubseteq)$ is a complete lattice (i.e., every subset of D has a least upper bound)
  - Every subset of D has a greatest lower bound
- **Corollary:** every complete lattice has a greatest element $\top := \bigsqcap \emptyset$

Chains

Chains are generated by the approximation of the analysis information in the fixpoint iteration.

Definition 3.10 (Chain)

Let $(D, \sqsubseteq)$ be a partial order. A subset $S \subseteq D$ is called an (ascending) chain in D if, for every $s_1, s_2 \in S$,

$$s_1 \sqsubseteq s_2 \text{ or } s_2 \sqsubseteq s_1$$

(that is, S is a totally ordered subset of D).

Example 3.11

1. Every $S \subseteq \mathbb{N}$ is a chain in $(\mathbb{N}, \leq)$
2. $\{\emptyset, \{0\}, \{0,1\}, \{0,1,2\}, \ldots\}$ is a chain in $(2^{\mathbb{N}}, \subseteq)$
3. $\{\emptyset, \{0\}, \{1\}\}$ is not a chain in $(2^{\mathbb{N}}, \subseteq)$

The Ascending Chain Condition

Termination of fixpoint iteration is guaranteed by the Ascending Chain Condition.

Definition 3.12 (Ascending Chain Condition)

A partial order $(D, \sqsubseteq)$ satisfies the Ascending Chain Condition (ACC) if each ascending chain $d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ eventually stabilizes, i.e., there exists $n \in \mathbb{N}$ such that $d_n = d_{n+1} = \ldots$

Example 3.13

1. $(\mathbb{N}, \leq)$ does not satisfy ACC
2. (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$ satisfies ACC since Var_c (unlike Var) is finite
3. (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$ satisfies ACC since $AExp_c$ (unlike $AExp$) is finite